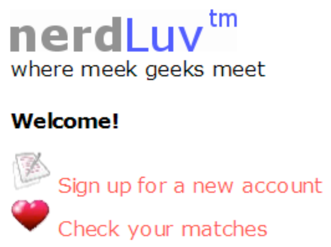
University of Washington, CSE 190 M
Homework Assignment 4: NerdLuv

This assignment is about making a simple multi-page "online dating" site that processes HTML forms with PHP. **Online dating** has become mainstream with popular sites such as eHarmony, match.com, OkCupid, Chemistry, and Plenty of Fish. Your task for this assignment is to write HTML and PHP code for a fictional online dating site for desperate single geeks, called **NerdLuv**. Turn in the following **four files**:

- **signup.php**, a page with a form that the user can use to sign up for a new account
- **matches.php**, a page with a form for existing users to log in and check their dating matches
- **index.php**, a page with links to signup.php and matches.php
- **common.php**, a file containing shared code used in your signup.php, matches.php, and index.php files

There are some **provided files** on the web site. The first is **skeleton.html**, an HTML file that looks identical to your final **index.php** page's HTML output. You will take apart this file to create your final index.php page as well as portions of your **signup.php** and **matches.php** files. We also provide a complete CSS file **nerdluv.css** with all of the page styles you should need. You should link to this CSS file from your HTML and make use of its styles in your code. You should be able to fully style all pages using the styles in **nerdluv.css** only.

Overall Site Navigation:

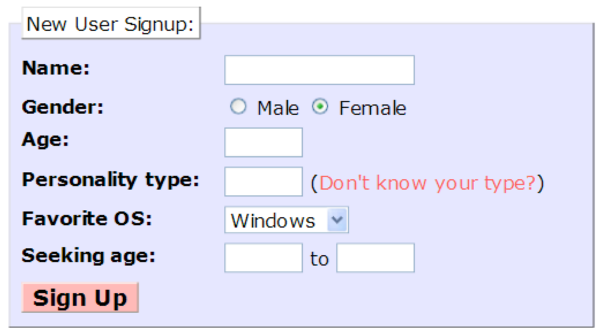

nerdLuv tm
where meek geeks meet

Welcome!

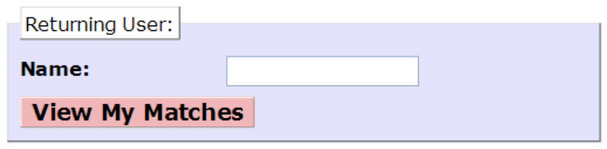
Sign up for a new account
Check your matches

Your dating site's Index Page (**index.php**, left) links to the two other content pages.

The "Sign Up" link leads to **signup.php** (left below), and "Check matches" to **matches.php** (right below):

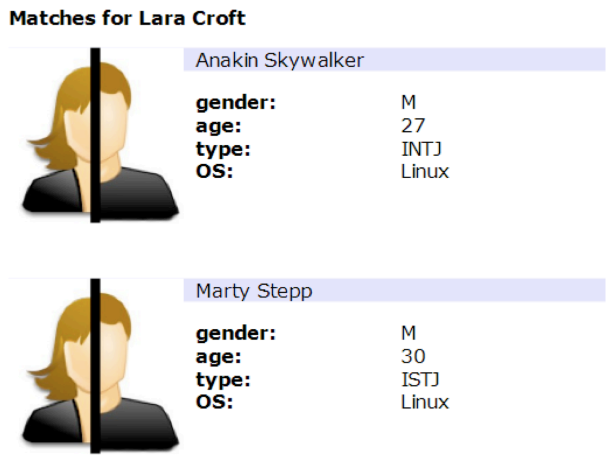

New User Signup:

Name:	
Gender:	Male [✓ Female]
Age:	
Personality type:	(Don't know your type?)
Favorite OS:	Windows
Seeking age:	to


Returning User:

Name:

When submitted, the View Matches form looks like this:


Matches for Lara Croft

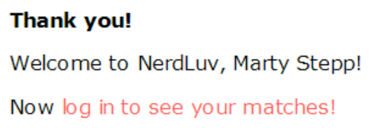
Anakin Skywalker

gender:	M
age:	27
type:	INTJ
OS:	Linux

Marty Stepp

gender:	M
age:	30
type:	ISTJ
OS:	Linux

When submitted, the Signup page looks like this:


Thank you!

Welcome to NerdLuv, Marty Stepp!

Now log in to see your matches!

The details about each page's contents and behavior are described on the following pages. Screenshots in this document are from Windows XP in Firefox 3.5, which may differ from your system.

Skeleton File (skeleton.html) and Shared Content (common.php):

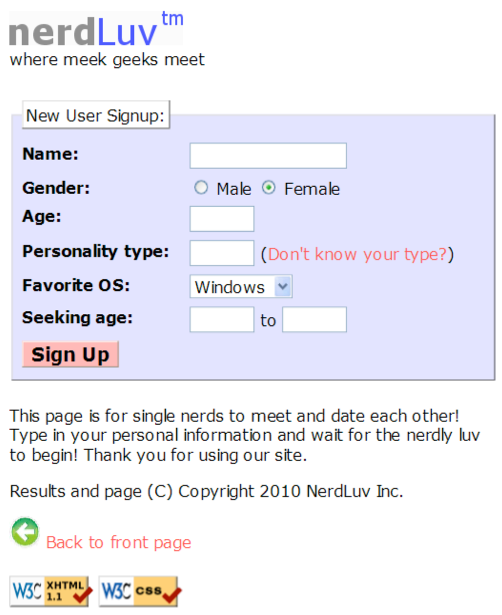
The site's three pages (**index.php**, **signup.php**, and **matches.php**) have a lot of content in common: they share an identical header, footer, validator images, and basic HTML skeleton (doctype, `html` tag, `head` tag, etc.).

The provided **skeleton.html** file's content is identical to the HTML output of your final index.php file. As such, it contains all the shared content—header, footer, etc.—that you need to build the other files, as well as the links that make up the main content of **index.php**. So you should take apart the content of the skeleton.html file and use it to build your final PHP pages.

However, rather than putting copies of the shared HTML code in each of your PHP files, you should factor out this redundancy by putting all shared HTML code into PHP functions in your **common.php** file and calling those functions from the other files whenever you need to output shared HTML code. For example:

```
<?php function print_top_html() { ... } ?>
```

Sign-Up Page (signup.php):



nerdLuv tm
where meek geeks meet

New User Signup:

Name:	
Gender:	○ Male ◉ Female
Age:	
Personality type:	(Don't know your type?)
Favorite OS:	Windows
Seeking age:	to

Sign Up

This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

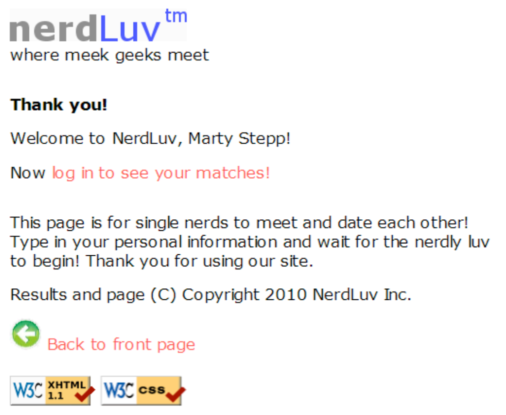
Results and page (C) Copyright 2010 NerdLuv Inc.

Back to front page

The **signup.php** page has a **form** to create a new account, and should output the same shared HTML content (header, footer, etc.) as the other files (see above). You must write the HTML code for the form. The form contains the following labeled fields:

- **Name:** A 16-character-wide box for the user to type a name.
- **Gender:** Radio buttons for the user to select a gender of Male or Female. When the user clicks the text next to a radio button, that button should become checked. The Female button should be checked initially.
- **Age:** A 5-letter-wide text input box for the user to type his/her age in years. The box should allow typing up to 2 characters.
- **Personality type:** A 5-character-wide text box allowing the user to type a Keirsey personality type, such as ISTJ or ENFP. The box should let the user type up to 4 characters. The label has a link to http://www.humanmetrics.com/cgi-win/JTypes2.asp .
- **Favorite OS:** A drop-down select box allowing the user to select a favorite operating system. The choices are Windows, Mac OS X, and Linux. The "Windows" option should be selected initially.
- **Seeking age:** Two 5-character-wide text boxes for the user to specify the range of acceptable ages of partners. The user should be allowed to type up to 2 characters in each box. Initially both should be empty.
- **Sign Up:** When pressed, submits the form for processing as described below.

Submitting the Sign-Up Form:



nerdLuv tm
where meek geeks meet

Thank you!

Welcome to NerdLuv, Marty Stepp!

Now log in to see your matches!

This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright 2010 NerdLuv Inc.

Back to front page

When the user presses "Sign Up," the form should **submit** its data using the POST method. (The exact names/values of the submission parameter(s) are up to you.) The form should submit the data back to **signup.php** itself. Your **signup.php** code should therefore have two modes:

- If the browser is performing a normal **GET** request, your page should display the signup form as described above.
- If the browser is **POST**ing data, your PHP code should read the data from the query parameters and store it as described

below. The resulting page has no form; instead, it has text thanking the user for signing up, again with the usual header and footer. The text "log in to see your matches!" should link to **matches.php**.

(Hint: In PHP, you can check for a GET or POST request with the variable `$_SERVER["REQUEST_METHOD"]` as described in Ch. 6 or the lecture slides.)

Form Data:

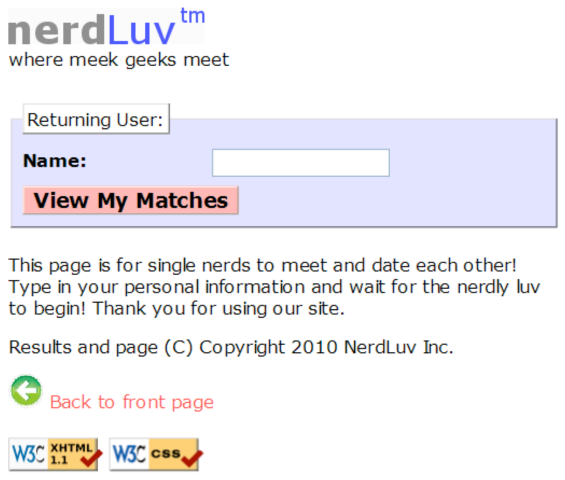
Your site's user data should be stored in a file **singles.txt**, placed in the same folder as your PHP files. We will provide you an initial version of this file. The file contains data records as lines in *exactly* the following format, with the user's name, gender (M or F), age, personality type, operating system, and min/max seeking age, separated by commas:

```
Angry Video Game Nerd,M,29,ISTJ,Mac OS X,1,99
Lara Croft,F,23,ENTP,Linux,18,30
Seven of Nine,F,40,ISTJ,Windows,12,50
```

Your **signup.php** page's POST response should create a line representing the new user's information and add it to the end of the file. *(Hint: See the PHP function `file_put_contents` in Ch. 5 or the lecture slides.)*

In all pages, **assume valid data** for the file's contents and form submissions. For example, no fields will be left blank or contain illegal characters (such as a comma). No user will attempt to resubmit data for a name already in the system.

View Matches Page (matches.php):



This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright 2010 NerdLuv Inc.

The **matches.php** page has a **form** to log in and view the user's matches, and should output the same shared HTML content (header, footer, etc.) as the other files (see above). You must write the HTML code for the form. The form has just one field:

- **Name:** A label and 16-character-wide box for the user to type a name. Initially it should be empty. When the form is submitted to the server, the parameter for this field should be called `name`.

When the user presses "View My Matches," the form **submits** its data back to **matches.php** itself, using the GET method. The exact name of the query parameter should be `name`, and its exact value should be the encoded text typed by the user. For example, when the user views matches for Rosie O Donnell, the URL should be:

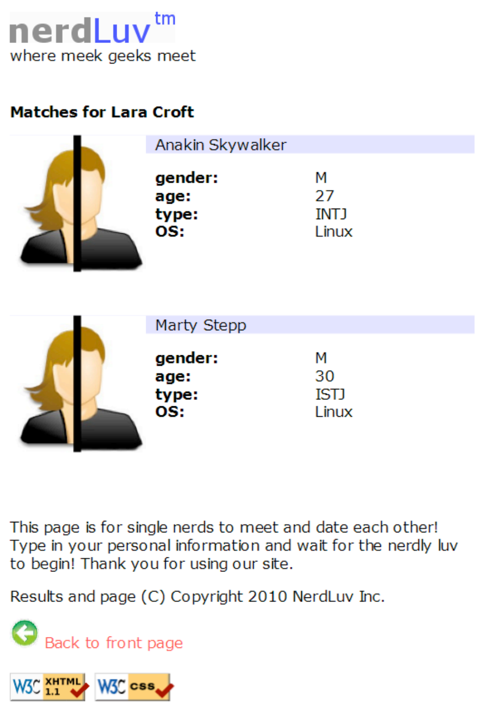
- **matches.php?name=Rosie+O+Donnell**

As with **signup.php**, this page operates in one of two "modes":

- If the browser is performing a GET request *without* a query parameter called `name`, it should display the "Returning User" form described above.

- If the browser is performing a GET request *with* a query parameter called `name`, the form should not be shown. Instead, your PHP code should search for, and output a list of, users that are "compatible matches" with the user with the given name, as described below.

Viewing Matches:



nerdLuv tm
where meek geeks meet

Matches for Lara Croft

Anakin Skywalker

gender:	M
age:	27
type:	INTJ
OS:	Linux

Marty Stepp

gender:	M
age:	30
type:	ISTJ
OS:	Linux

This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright 2010 NerdLuv Inc.

Back to front page

When viewing matches for a given user, your matches.php page should show a central area displaying a list of compatible matches. Write PHP code that reads the user's name from the page's query parameter and figures out which other singles are compatible with the given user. The existing singles to match against are records found in the file **singles.txt** as described previously. You may assume that a record exists in the singles.txt file for the submitted user.

Below the page header should be a heading saying "Matches for *(name)*". Below this should be a list of singles that are compatible matches with the given user. A "match" is a person from the data set with the following qualities:

- The **opposite gender** of the given user;
- Of **compatible ages**; that is, each person (i.e., both the returning user *and* the potential match) is between the other's minimum and maximum ages, inclusive;
- Has the **same favorite operating system** as the given user;
- Shares **at least one personality type letter in common**. For example, types ISTP and ESFP have two in common (S and P).

As you find each match, you must emit the HTML to display that match. Matches should be listed in the same order the matching people were found in the file. Each match shows the image **user.jpg**, the person's name, and an unordered list with their gender, age, personality type, and OS.

http://www.cs.washington.edu/education/courses/cse190m/11sp/homework/4/user.jpg

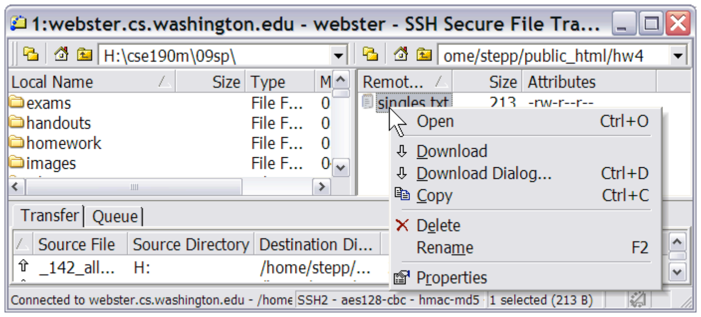
Styling:

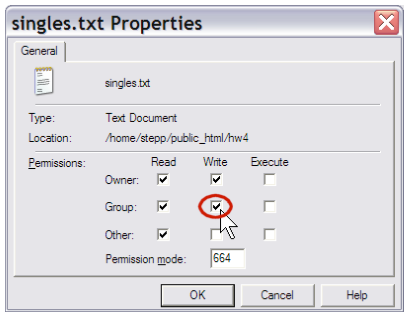
The styles you need are already given to you in **nerdluv.css**, but you will need to inspect that file to see which tags and `class` attributes to choose to make sure the style rules it contains are applied to your content. Be mindful of the styles on forms and form controls. On the course web site are several screenshots of the various pages. Make sure that your form has the same width, colors, fonts, borders, etc. as in these examples. If you choose reasonable tags to represent your page content, your output should match exactly.

In the list of matches on **matches.php**, each match is displayed in a `div` with `class` of `match`. First is a paragraph containing an image of the match, shown with a width of 150px, and the person's name to the right. The paragraph has a light blue background color. The section with the match's gender, age, etc. can be represented either as an unordered list (`ul`) or as a definition list (`dl`).

Uploading and Testing:

Upload all files to Webster to test them. You may need to change the **permissions** on **singles.txt** so that PHP can write changes to it. In SSH Tectia, right-click **singles.txt** in the right pane and choose Properties. Enable Group Write by checking the box shown below.





singles.txt Properties

General

singles.txt

Type: Text Document
Location: /home/stepp/public_html/hw4

Permissions:	Read	Write	Execute
Owner:	✓	✓	
Group:	✓	✓	
Other:	✓		

Permission mode: 664

Suggested Development Strategy and Hints:

- Turn the provided **skeleton.html** file into **index.php**, then use this as a basis to write **matches.php**, so that your site works properly for existing users. Don't worry about redundant shared HTML between these files for now.
  - Write an **initial version** that outputs *every* person, even ones who aren't compatible "matches." This way you can debug your file I/O, styles, etc. Then add checks like gender, age, and OS. Focus on the PHP code and behavior first, as opposed to style details (CSS is not worth many points on HW4).
- Write **signup.php**. If you finished the match page you'll understand forms, making the signup page easier. This page is tough; there are more form parameters to manage, and you must write to the file.
- Factor out all shared HTML from **index.php**, **matches.php**, and **signup.php** into functions inside **common.php** that are called from the other files whenever common HTML content needs to be output.

Use **debug `print` and `print_r` statements** to track down bugs. For example, you can `print_r($_REQUEST);` to see the query parameters submitted. Use **Firebug** and also **View Source** to find HTML output problems.

With the exception of the submit button, all form controls must have `name` attributes. Sometimes you must also add a `value` to affect how data is submitted. You can test a form by setting its `action` attribute to Webster's `params.php`, which prints all the data that was submitted to it for debugging purposes.

Implementation and Grading:

Your HTML output for all pages must pass the W3C **XHTML validator**. (Not the PHP source code itself, but the HTML output it generates.) Do not use HTML tables. Since we are using HTML **forms**, choose proper form controls and set their attributes accordingly. Properly choose between GET and POST requests for sending data.

Your PHP code should not cause errors or warnings. Minimize use of the `global` keyword, use good indentation/spacing, and avoid lines over 100 characters. Use material from the first four weeks of class and the first six book chapters.

A major grading focus is **redundancy**. Use **functions, parameters/return, included files/code**, loops, variables, etc. to avoid redundancy. Some sections of HTML code are identical between all your PHP pages. These redundant sections should be printed by functions in your common.php file. Include this file in all of your other pages using the `include` function to make use of its contents.

For full credit, reduce the amount of large chunks of PHP code in the middle of HTML code. Replace such chunks with **functions** declared at the top or bottom of your file. You will also lose points if you use PHP `print` or `echo` statements. Insert dynamic content into the page using PHP **expression blocks**, `<?= ... ?>` , as taught in class.

Another grading focus is PHP **commenting**. We expect more comments here, similar to CSE 14x. Put a descriptive comment header at the top of each file, **each function**, and each section of PHP code.

Format your HTML and PHP code similarly to the examples from class. Properly use whitespace and indentation. Do not place more than one block element on a line or begin a block element past the 100th character. Do not allow a line of PHP code or comment to extend past the 100th character.

Our solution uses this many lines: **signup.php** 80 (56 "substantive"); **matches.php** 80 (61 "substantive").

Please do not place a solution to this assignment online on a publicly accessible web site. Part of your grade will come from successfully uploading your files to the **Webster** server in the directory:

- **https://webster.cs.washington.edu/*your_uwnetid*/hw4/**